AC Current Generator (ACCG)

by Aubrey Kagan & Ernesto Gradin

Introduction

Weidmuller manufactures a wide range of industrial signal conditioners. Among them is a series of modules that translate AC current to a standard control signal of 4-20mA or 0-10V. Integral to the manufacturing process is calibration and test necessitating the generation of an AC current to stimulate the input of the module under test.

The component structure of the project allow for any reasonable current to be generated, and since the control is hosted on a PC it is possible to configure any kind of user interface. Since we have no intention of making this into a commercial product, all the relevant information has been supplied.

The currents that need to be generated for our products can, in some cases, exceed 100Amps AC. As a result, the use of resistive techniques was discounted as they generate excessive heat. In addition, they are also potentially dangerous to the test technician since there could be 120V present at the module inputs. A further disadvantage is that the test is restricted to the line frequency and subject to the unpredictable changes of the line voltage.

A more common solution in the industry is to use a toroidal transformer with only a few turns on the secondary. For a transformer, the power presented at the primary is equal to the power delivered at the secondary, and the output voltage is proportional to the turns ratio of the secondary winding to the primary winding. It is possible by using these two properties to construct a transformer to generate a high current at a low output voltage.

Initially we tried this with a variac driving the primary, but found that the line voltage variations were too great to be practical. In addition the output frequency was still limited to the line value.

The next attempt was to use a sine wave oscillator driving a power amplifier that was connected to the primary of the transformer. In this configuration it was possible to change the AC frequency, but it suffered from another problem. The transformer secondary current is affected by the loop resistance which includes the terminal resistance of the module and the wire length. For each module calibrated the gain setting of the oscillator would have to be changed, resulting in a lengthened calibration time.

Obviously the solution was to create a closed loop system. Once implemented, additional features become available such as surge testing and step responses. We decided not only should we design a controller to realize this goal, we should also use the project as a vehicle to learn new technologies. Some of the goals were to learn Accel (schematic capture and layout), Visual Basic and a software based PID control loop.

Features

The ACCG can produce an AC current in the form of several processes:

IDLE- the output is off
OPEN LOOP- the output current is set to a value and left unchanged thereafter
CLOSED LOOP- the output is adjusted in an attempt to keep the measured signal at channel 0 constant.
STEP- a combination of the IDLE and OPEN LOOP processes in which the output is turned on and off for predetermined periods for a predetermined number of times.

It is possible to generate other outputs through manipulation of the ACCG by the software on the host PC. Functions such as a ramp may be achieved. Associated with processes are a number of parameters detailed in Table 3. These parameters are saved in the EEPROM of the ACCG and can be read or changed as desired.

The host software is written in Visual Basic so that it may be easily customized for any application without changing the firmware which requires detailed knowledge of the operation of the hardware.

System Overview

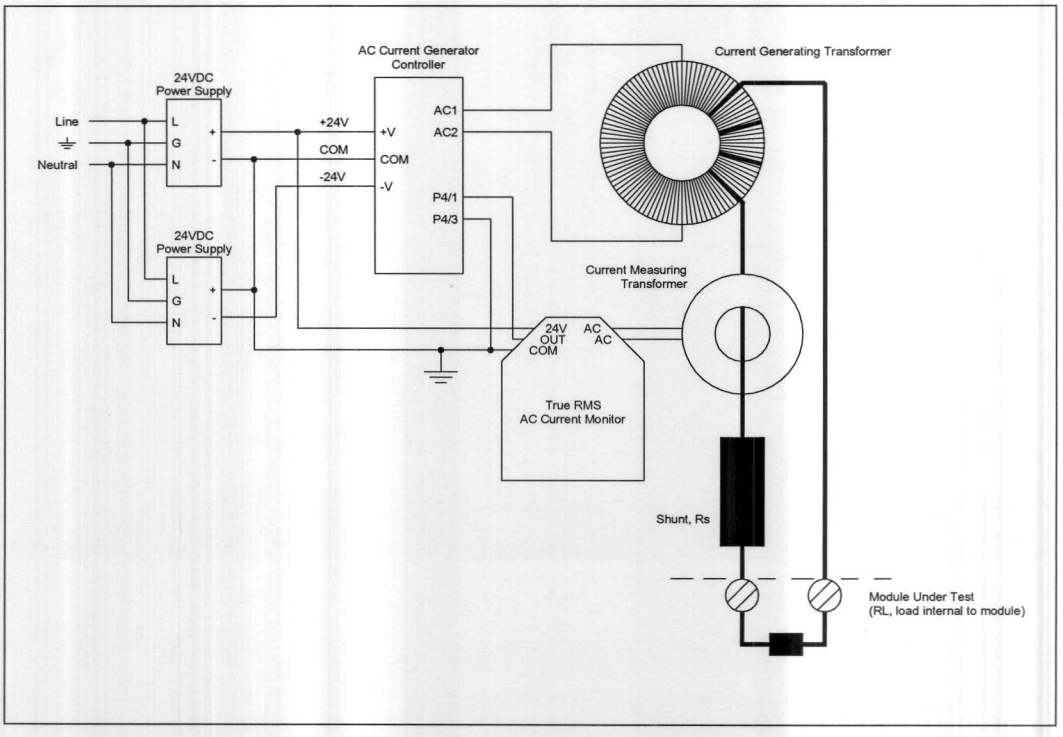


Figure 1- System Overview

Power for the AC Current Generator controller is derived from two 24VDC power supplies. They are connected in series to generate a positive and negative supply. The positive supply also provides power for the True RMS AC Current Monitor. Based on the work we did for 10Amp and 100 Amp outputs, 60 Watt power supplies are all

that are really necessary. Since Weidmuller produces a wide range of switchmode power supplies, our choice of manufacturer was obvious.

The AC Current Generator controller is a module developed especially for this project. Its component parts and functions are described later. In essence it is a microcontroller, A/D converter, AC waveform generator and power amplifier. The AC output from the power amplifier drives the primary of the Current Generating Transformer. The Current Generating Transformer must be wound for a particular current range. Details of this are described in a following section.

The secondary of the Current Generating Transformer is used to drive the Module that is being tested. The secondary loop passes through a shunt (for calibration purposes) and the primary of a Current Measuring Transformer (single turn!) before being closed by the load itself, RL. The output of the Current Measurement Transformer is converted to a 4-20mA signal through a True RMS Current Monitor. By some coincidence, Weidmuller manufactures several suitable products. We selected the Bicron B5303 current transformer to drive the Weidmuller 991064 True RMS AC Current Monitor module. The 4-20mA signal can then be processed by the AC Current Generator controller. Calibration is not really an issue with this module as the system software is arranged to allow a current to be generated in an open loop, and then the corresponding output as measured by the AC Current Generator controller A/D converter is "learned" . The closed loop control is executed around that value.

In order to calibrate the unit (during the "learn" process) the actual AC current must be known. This is achieved by measuring the RMS voltage across the Shunt, Rs. From Ohm's Law it is possible to calculate the RMS current in the loop.

Magnetics

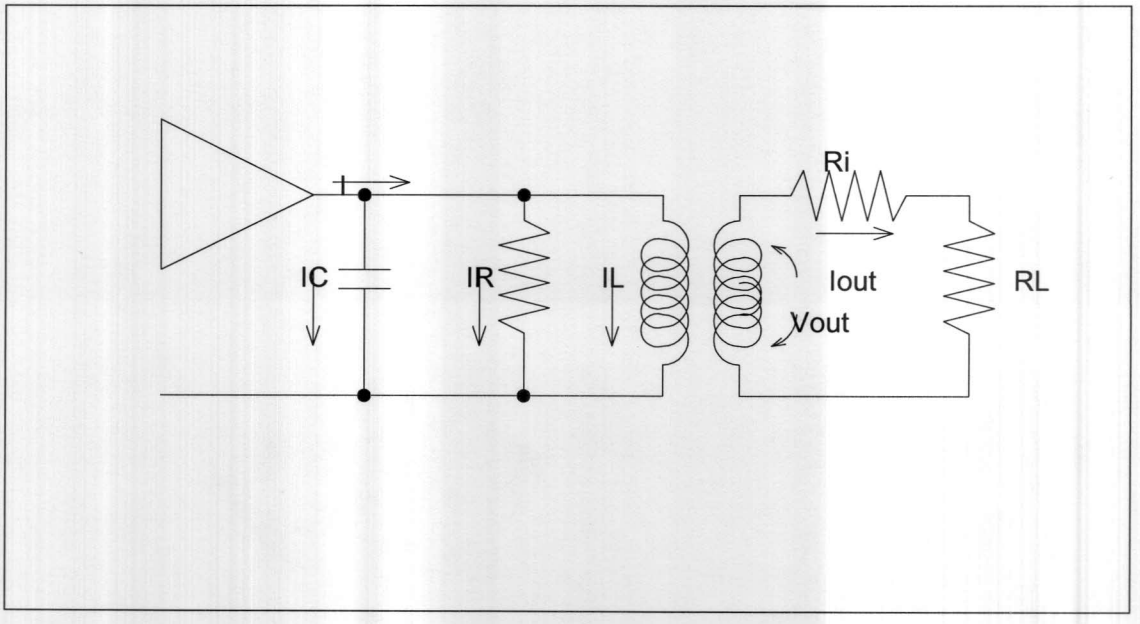


Figure 2-Equivalent Circuit

Figure 2 is the equivalent circuit of the Current Generating Transformer. The output current from the power amplifier has three components of three different phases. I_L is the current through the inductive component, I_R is the current through the reflected resistive component (from the secondary) and I_C is the current through the capacitance of the circuit. A capacitor can be added to cancel the current required by the inductive component, reducing the current drive requirements of the amplifier.

The internal resistance of the secondary R_i is made up of wire resistance plus the transferred impedance of the primary plus the resistance of the measurement shunt and the transferred impedance from the secondary of the Current Measuring Transformer. R_L is the load in the unit under test.

Step 1- Establish Constants

In order to design the transformer, we must introduce values from the actual circuitry that we are going to use. For this example we decided to generate an output current of 10Amps AC. The load R_L is 25 milliOhms (a shunt internal to the product), and the wiring resistance of the secondary, R_i is assumed to be 25 milliOhms as well.

In addition, a transformer core must be selected. We had several spare toroidal transformers which we proceeded to disassemble. The measurements of the transformer core were as follows:
Cross–sectional height H=2.15 cm
Cross-section width W=1.60 cm
Inner diameter D= 3.93 cm

Side bar- Finding the Inductance per turn^2 by calculation

L (in Henry) $=(1.256*S*N^2*\mu e*10^{-8})/C_m$
Where S is the core cross section in square cm, C_m is the mean circumference of the toroid ($2*\prod*$(outside diameter-inside diameter)/2), and N is the total number of turns. μe is the effective permeability of the material. In a toroid this is essentially the permeability of the material. Once L has been calculated from the above formula, L_T can be simply calculated as follows:

$L_T = L/ N^2$

Prior to the other calculations the inductance per turn L_T must also be known. This can be calculated from the magnetic constants if the core data is available (see sidebar). Alternatively it can be measured with an LCR bridge. Winding 5 turns through the core and measuring the inductance gave 40.7uH. The inductance is proportional to the square of the number of turns. Therefore L_T is calculated as $40.7/5^2 = 1.63$uH/turn^2.

Step 2- Primary Turns

Magnetic theory predicts the following relationship for the primary of the transformer:

$E=4.44*f*N_P*S*B*10^{-8}$

Where E is the RMS voltage, f is the frequency, N_P is the number of turns on the

Sidebar: Maximum Allowable Flux Density

The maximum allowable flux density before saturation of an iron core is 12000 to 15000 gauss. 8000 to 10000 gauss is a safe value. Iron cores can be used up to a frequency of 400Hz. For higher frequencies a ferrite core must be used. In this case the flux density must be reduced to no more than 2500 to 3000 gauss to prevent saturation of the core.

primary, S is the cross sectional area of the core, and B is the flux density in the core (in gauss).

We can re-arrange this to find the number of turns:

$N_P = (E*10^8)/(4.44*f*S*B)$

In our setup E=14Vrms. This is derived from the supplies being ±24V. Allowing for 4V "headroom" for the power amplifier, this gives V_{pp} of 40V.
$V_{rms} = V_{pp}/(2*sqrt(2))=E$.
The frequency, f, is nominally 60Hz.
The cross sectional area, S is equal to H*W=2.15*1.60=3.44 cm^2
The constant B is approximately equal to 8000 gauss (????).
Substituting these values

$N_p = (14*10^8)/(4.44*60*3.44*8000) \cong 190$ turns

This is the number of turns required to maintain the core below saturation at the maximum input voltage.

Step 3- Secondary Turns
The next step is to calculate the number of turns on the secondary, Ns. As can be seen from **Figure 2** and based on Ohm's Law:
Vout = Iout*(Rs+Ri).
Since we have chosen the output to be 10Amps and Rs=25milliOhms and Ri=25milliOhms,
Vout=10*0.05=0.5Vac.

The turns ratio is the proportion of the input voltage to the output voltage

n= E/Vout=14/0.5 = 28.

Therefore the number of turns on the secondary N_s:

$N_s = N_p/n = 190/28 = 7$.

Step 4- Primary Winding

The length of wire required for the primary is given by the perimeter of the cross section multiplied by the number of turns. For our case the perimeter is (2*2.15)+(2*1.60)=7.5 cm. The length of wire is 190*7.5= 1425cm ≅15 metres.

In order to choose the correct wire for the primary we need to consider the inside circumference of the toroid. **Figure 3** describes the basis of the calculation. The interior diameter of the core is 39.3 mm. The internal circumference is ΠD = 3.14*39.3 ≅ 123 mm. The maximum diameter of a wire that will fit into this circumference is 123/190 ≅ 0.6 mm with one layer. More than one layer is permitted. Using wire tables magnetic wire of size AWG22 is used (0.71mm with insulation).

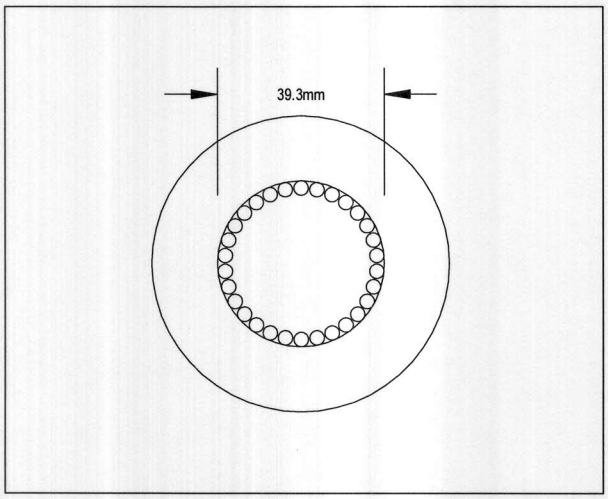


Figure 3- Primary Winding Wire Thickness Calculation

The resistance of the primary R_p is the product of the wire resistivity (0.53 miiliOhms/cm for AWG22 copper) and the wire length

$R_p = 0.00053 * 1500 \cong 0.8$ ohms.

The reflected resistance to the secondary is $R_p' = R_p/n^2 = 0.8/28^2 \cong 1.0$milliOhm

Step 5- Secondary Winding

Based on wire tables we need a wire gauge of at least AWG#12 for 10Amps. It has a resistivity of 0.052 milliOhms/cm. Allowing for 100cm of wire (only about 52 cm is needed for the actual loops) the wire resistance R_s, is 5.2 milliOhm. The total internal resistance R_i is the sum of R_p and R_s i.e. 5.2+1 = 6.2 milliOhm.

If R_i plus the sum of the addition resistance of the wire and contact resistance of the module under test exceed the 25 milliOhm that we selected at the start, the 14 Vrms that we use to drive the primary would have been insufficient. Increasing the primary voltage would increase the flux density and the core would saturate.

Step 6- Primary Inductance

The primary inductance is the inductance per turn multiplied by the (number of turns)^2 .

$L_m = N_s^2 * L_T = 190^2 * 1.63uH/turn^2 = 58.8mH \cong 60mH$. The magnetic (reactive) current I_L (see **Figure 2**) flows through this. The impedance is given by $X_L = 2*\Pi*f*L_m = 2*3.14*60*60mH \cong 22$ ohms

Step 7- Primary Reflected Impedance

The secondary load of nominally 50 milliOhm is reflected to the primary $R_s' = R_s * n^2 = 50$ milliOhm $* 27^2 \cong 36$ ohm. The current I_R (see **Figure 2**) flows through this resistive load.

Step 8- Amplifier Drive Current

If the capacitance X_c is purely parasitic, the current through the capacitance is negligible so the total current is the vector sum of IL and IR. This is

$I = \text{sqrt}(I_L^2 + I_R^2) = \text{sqrt}(E^2/X_L^2 + E^2/R_L^2) = E\ \text{sqrt}(1/X_L^2 + 1/R_L^2) =$
$= 14\ \text{sqrt}(1/22^2 + 1/36^2) \cong 750$mA. The amplifier must be capable of delivering this. In addition the power supplies must be capable of sourcing and sinking this current as well as supplying the other needs of the circuit.

If a capacitor is added it should be equal to the inductive component

$X_C = X_L = 1/(2 * \Pi * f * C)$ where f is the frequency and C is the capacitance. From above this is 22 ohm

$C = 1 / (2 * \Pi * f * X_C) = 1 / (2 * 3.14 * 60 * 22) \cong 120\mu F$.

This capacitor must be non-polar. It can be achieved by back to back electrolytic capacitors of double the value. Since the amplifier is not overtaxed by the uncompensated requirements, we chose not to use the capacitor. Of course at higher frequencies the value of the capacitor is reduced.

Step 9- Maximum Possible Current

Based on Ohm's Law the secondary current Is can be expressed as follows:

$I_s = E_p/(n*R_s + R_p/n)$

Where E_p is the maximum voltage across the primary, n the turns ratio N_p/N_s, R_s is the secondary resistance and R_p the primary resistance. Through some painful calculus it is possible that to show that a maximum current (for given resistances) is achieved when $n*R_s = R_p/n$. Solving this for n we get

$n = \text{sqrt}(R_p/R_s)$

Using our figures to date, $n = \text{sqrt}(0.8/0.05) = \text{sqrt}(16) = 4$. With an N_p of 190 N_s becomes 47. From above

$I_s max = E_p/(4 * R_s + R_p/4) = 14 / (0.2 + 0.2) = 35$ amps!

However with the maximum number of turns in the secondary, the secondary resistance will go up, but we can see that there is still quite a bit of "headroom" and it is possible to get a higher current simply by increasing the number of windings.

Sidebar- Transformer Calculation for 100 Amps

On our products that use the 100 Amp input, there is no input shunt resistor. The current is sensed through the Bicron current transformer. We also wanted to used the same transformer core as before.

Step 1- Establish Constants
The shunt resistor (for calibration) in the secondary of the transformer is 0.5milliOhms. If we manage to limit the resistance of the loop to 1 milliOhm, the power dissipated across the secondary of the system is (I^2R) at 100 Amps is 15W. This can translate to quite a bit of heat.

The Bicron B5303 Current transformer has a current turns ratio of 5000 and a wire resistance of 207 Ohms. The output is connected in series with a 10 ohm resistor in its secondary (internal to the measurement device and the device under test. (There are 2 such devices in the loop.) The reflected impedance for each is $217/5000^2 = 9$ μOhm. So this should not pose a problem.

Since we are using the same transformer core The dimensions do not change.

Step 2- Primary Turns
We use the same result N_p=190 turns.

Step 3- Secondary Turns
Let us assume that we can limit the resistance to 1.5millOhms,

Vout = Iout*(0.0015)= 100*0.0015= 150 milliVolt and so the turns ratio:

$n = E/Vout = 14/0.15 \cong 93.3$

The turns on the secondary $N_s = N_p / n = 190/93.3 \cong 2$ turns.

Since the wire size is going to need to be fairly thick (see step 5), the fewer turns there are, the easier it will be to wind the transformer and the lower the series resistance.

Step 4- Primary Winding
This produces the same selection of wire and primary resistance as previously. The reflected resistance $R_p' = R_p/n^2 = 0.8/93.3^2 \cong 92$ μOhm.

Step 5- Secondary Winding
For 100 Amps we need a wire of diameter 5mm or thicker. We opted for the battery strap of a car battery. It was 25" long, 4 gauge, and already made up with connection tags. Since there are no high voltages, isolation of the insulator is not an issue.

Step 6- Primary Inductance
As before X_L=22 ohms.

Step 7- Primary Reflected Impedance
The secondary load of nominally 1.5millOhm is reflected to the primary $R_s' = R_s * n^2 = 0.0015 * 93^2 \cong$ 13 Ohms

Step 8- Amplifier Drive Current
$I = \text{sqrt}(I_L^2 + I_R^2) = \text{sqrt}(E^2/X_L^2 + E^2/R_L^2) = E\ \text{sqrt}(1/X_L^2 + 1/R_L^2) =$
$= 14\ \text{sqrt}(1/22^2 + 1/13^2) \cong 1250$mA.

Step 9- Maximum Possible Current
From the previous derivations:
$n = \text{sqrt}(R_p/R_s) = \text{sqrt}(0.8/0.0015) = \text{sqrt}(533) = 23$

With an N_p of 190 N_s becomes 8. From above

$I_s max = E_p/(23 * R_s + R_p/23) = 14 / (0.0345 + 0.0345) = 202$ amps!

However with the maximum number of turns in the secondary, the secondary resistance will go up resulting in more heat, but we can see that there is still quite a bit of "headroom" .

Hardware

Figure 4, sheets 1 to 4 are the complete schematic of the AC Current Generator Controller.

The microcontroller is of the 8051 family. The only requirements are that it has 256 bytes of Random Access Memory, a serial port and sufficient program memory for the firmware, which is currently at 7K. We selected the Atmel AT89C55 simply because with 20K of EPROM there would be plenty of room for expansion. The controller is hardly in need of graceful recovery from a soft error, but for the sake of consistency with other designs an external watchdog is employed, both for its watchdog and reset capabilities. The crystal frequency for the microcontroller (11.0592MHz) is selected to provide a convenient divisor for a standard baud rate. The RS232 level shifting is achieved through the industry standard MAX232. The 5V supply is derived through a simple linear regulator from the +24VDC input. Originally the project was going to be a stand alone instrument, so an EEPROM was added to maintain the constants. This functionality was retained despite the fact that it can all be stored in the PC and downloaded. All peripheral devices use serial interfacing to simplify the printed circuit board.

The AC waveform source is a dedicated chip, the Micro Linear ML2037. This device is a sine wave generator with programmable frequency and gain with the output being centered around 2.5V. The frequency is based on dividing down a crystal oscillator, and the frequency was selected to give integer values for the oscillator. The device also has a shutdown input that allows the output to be turned on and off and always will start at the same wave phase which is a desireable feature for output step waveforms.

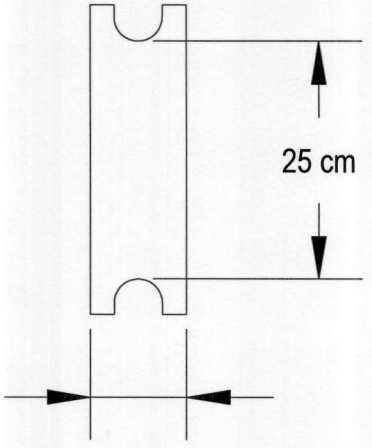
The output of the sinewave generator is passed through a programmable resistive divider, the DS1267. This allows for 512 discrete values, a resolution of 0.2%. Based on the results of the control loop, if I were to rework this project this is the area I would focus on.

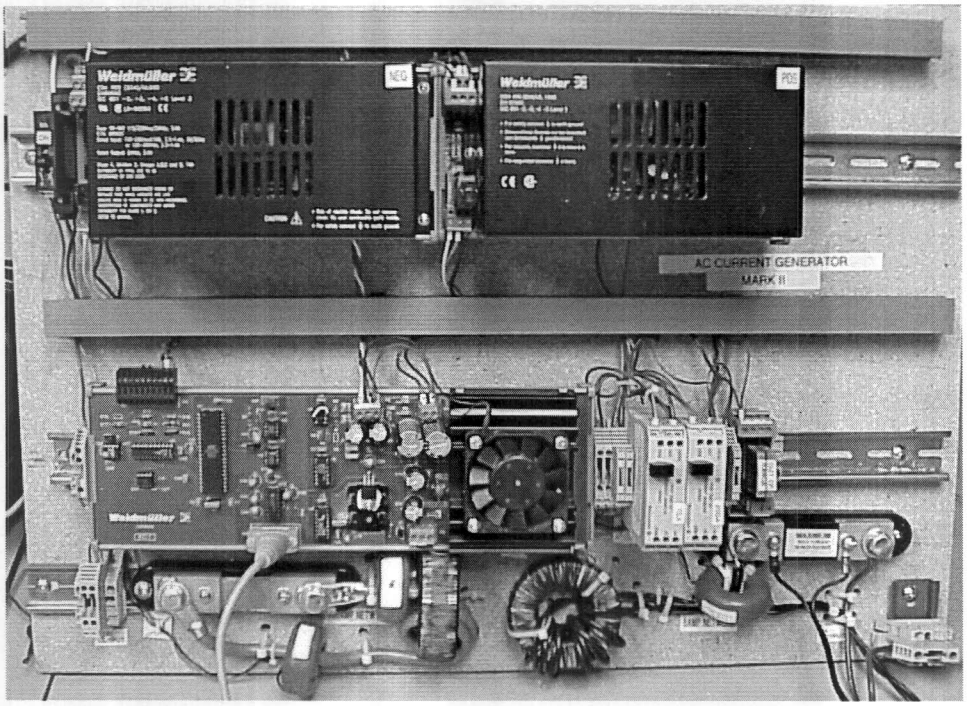
The wiper of the digital potentiometer is AC coupled to the input of the power amplifier, the LM12 from National Semiconductor. The op amp has a fixed gain of 1+(R7/R6) chosen for the maximum input (+/- 1V) to reach +/- 20V allowing for the "headroom" of the op-amp. The output of the op-amp drives the external transformer directly, but the return of the transformer is AC coupled through C18 and C19 to prevent any DC magnetizing current. The return transformer wire is biased to 0V by the resistive divider R9,R10 whose values are high enough to make a negligible contribution to the DC current.

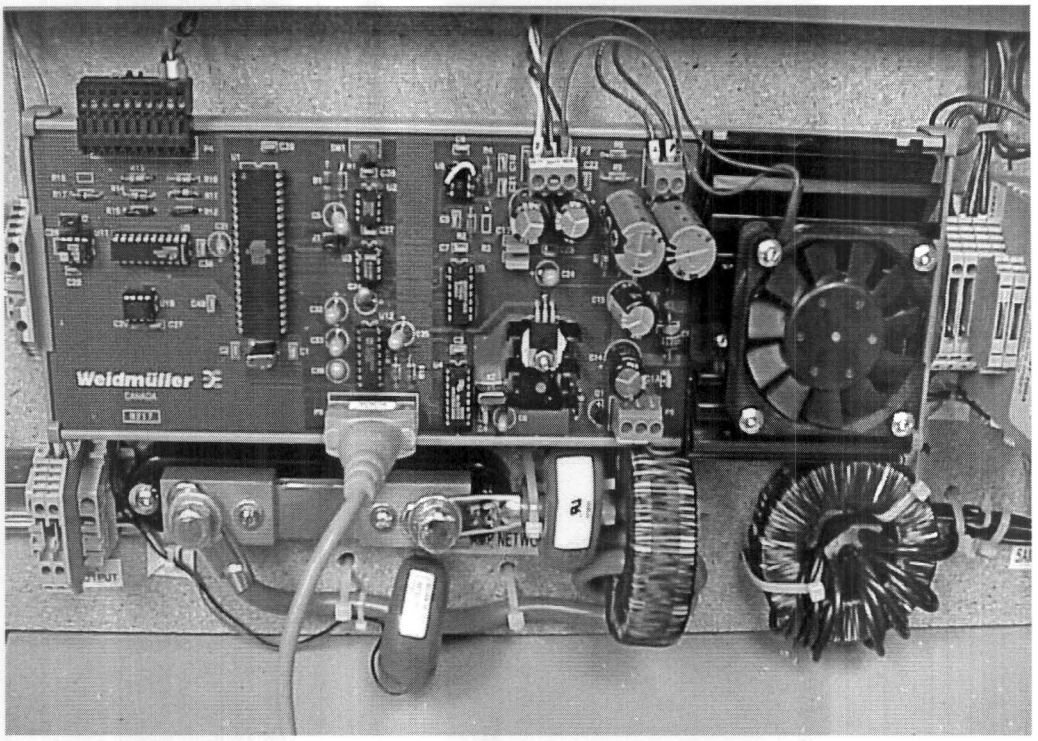
The returned analog signals are measured by a 12 bit A/D converter from Linear Technologies. The conversion from 4-20mA is simply achieved through a resistor to convert the current to a voltage. This resistor need not be accurate since the firmware will allow the system to "learn" the voltages associated with specific AC currents. Additional channels are available for other purposes so that the host PC may use them for any other purpose e.g. to generate a test procedure.

Sidebar- Practicalities of Winding a Toroid

Winding 190 turns of 22AWG wire (15 metres) onto a toroid can be a messy and frustrating affair. It is easier to pass a bobbin through the toroid than to wind 15 metres of loos wire. It is possible to make a bobbin as shown in the diagram, out of plastic or plywood. Wind the desired length of wire onto the bobbin, each full "loop" making ½ metre. The width of the bobbin (or the thickness of the wire spooled onto the bobbin) obviously must be less than the inner diameter of the toroid with its windings.




Weldmüller
NEG
Weldmüller
POS
AC CURRENT GENERATOR
MARK II
Weldmüller


Weidmüller
CANADA

Firmware

The embedded software for this project was developed in IAR C with occasional calls to assembler. The software was developed in modules as follows:
genmem.h, gen.h, genmem.c, geneep.asm, genback.c, gendflt.c, gentick.c, gen_ac.c, gen_ad.c, genPID.c, gencheck.c, gensint.c, genser.c
These modules are attached.

In order to create a form of multitasking, different tasks are created. In modern day terminology, these would probably be referred to as "objects". I have always referred to them as "phases". Each phase may broken down into further "sub-phases", and some may even employ "sub-sub-phases". The software is divided into a background routine, phase routines and interrupt routines. The background routine maintains a phase counter and for every loop of the background the counter is incremented so that a different phase is executed. This technique hands back processing to the background manager when it is ready to do so, rather than the classical technique of switching tasks based on a fixed time interrupt. The software is written so that the execution time in each phase does not hog too much of the execution time.

Since this is a simple application there are only 5 tasks. "checkA" and "checkB" form part of the error checking mechanisms of the software. Once again they are not really needed as part of this application, but the style persists from many other applications. "ad_convert" contains all the routines necessary to interface to the analog to digital converter. "ac_process" is the heart of the application where the ac current output is generated and controlled. "serCtl" contains the serial communications interface. Handshaking between the tasks is achieved by means of flags.

I hope that the listings are sufficiently commented to allow easy understanding. The following descriptions are intended as a high level viewMore detail is covered in the comments in the listings (I hope!).

Checks
The operation of the ACCG is based on the premise that if any "soft" error should occur, the watchdog timer will time out and force a reset. Further, if any error is detected by the software, rather than correct the error, the system goes into a closed loop thereby forcing a watchdog reset.

Optimally the watchdog should be clocked from only one place in a program and access to that location should be protected to prevent a runaway program from continuously clocking the watchdog, preventing the reset. Most software (certainly this application) has an asynchronous segment and an interrupt driven segment. In order to improve the probability that the watchdog will be clocked correctly, the watchdog strobe signal should only be set in the asynchronous part, and cleared during the interrupt sequence.

The checks used are implemented so that the background (asynchronous) process checks that the periodic interrupt routine is occurring, and the periodic interrupt routine ensures that the background is running. Additional safeguards are provided at the beginning of each phase to ensure that the phase number (used in the phase selection in the background) matches the actual phase being executed. On returning

from a phase the stack level is checked to ensure that it is correct. The references contain detailed explanations of this type of implementation.

Analog to Digital Conversion
Three a/d channels are read. When all 3 have been read they are transferred to a register and a flag set. Averaging is only done on channel 1, used for the 4-20mA signal from the True RMS Current Monitor.

Generate AC Current Process
Most of the functions and their implementation are self explanatory. On reset, the parameters stored in the EEPROM are read and some are loaded into RAM. The process number is one of these. On command it can be changed from the host PC.

The closed loop process is based on a PID loop. Provision is made for suitable integral, differential, and proportional constants. In reality it was found that the system worked best as a purely proportional loop. This was attributed to the long settling time and filtering effect of the external AC to mA converter coupled with the coarse AC current adjustment of only 9 bits.

Serial Communications
The serial communications portion of the software implements the Intercommunications Protocol as presented below.

EEPROM
Parameters are stored in EEPROM. The original software was developed in assembler. It was then ported to applications in C and so there are simple translation routines to match the used registers to the calling conventions used in IAR C. Only a small portion of the EEPROM is used for the application. These locations are protected by a checksum to detect whether the unit has been set up. It is possible to access all the available locations in the EEPROM for any purpose by following the communications protocol.

Most constants are only stored in EEPROM and are read from EEPROM when needed. A few are read and stored in RAM so that they can be changed "on the fly" by the host PC when the change in operations is desired without having to go through an EEPROM change. These include the process type and the target output for closed or open loop operation.

As part of failure protection an EEPROM read must provide two identical reads for the procedure to continue. Failure of the two subsequent readings to agree in three attempts will lead to a hardware reset occurring.

Communications Protocol

This is adapted from a standalone specification document.

1. The host PC may send commands to change parameters/operation and to read the status of the ACCG.
2. The ACCG will respond to every message from the PC. The response will depend on the request.
3. The transmission is full duplex, 9600 baud, 8 bits, no parity, and fixed message length of 32 bytes.

4. The first character is $3A_{hex}$.
5. All other characters are hex "nibbles" expressed in ASCII. Outside of the first and last characters, and command character, the only permissible characters are 0-9 and A-F (30_{hex} - 39_{hex} , and 41_{hex} to 46_{hex}). Every ACCG data byte results in 2 transmission bytes. The most significant nibble is transmitted first and the most significant byte is also transmitted first.
6. The checksum consists of the sum of all the ASCII bytes to be transmitted from byte 0 to byte 27. The resulting 16 bit total is then added one nibble at a time in ASCII as the last 4 bytes of the transmission. The most significant byte is first.
7. The performance and operational characteristics are determined by a number of parameters. The PC controls the operation of the ACCG by changing these parameters. Some of these are stored in EEPROM and will be retained with the cycling of power. Table 3 details the volatile parameters. Table 3 details the EEPROM parameters along with the associated addresses.
8. The PC can read back these parameters.
9. The ACCG may execute one of several processes as a current generator. These are listed in table 2.
   9.1. IDLE. Turn the output off and await a change in the process. It is still possible to read back the analog input signals.
   9.2. OPEN LOOP. This sets the output current register to the value contained in the command message. No attempt is made to control the output.
   9.3. PID. The output current is controlled so that the feedback signal on channel 0 of the A/D is held constant. This signal is a 4-20mA signal proportional to the AC current flowing in the loop.
   9.4. STEP. This will set the output current to a particular (programmable) value, hold it there for a time (programmable), change the current to a different value (programmable) and hold it there for a further (programmable) period. This process may be repeated continuously or for a finite number of times (programmable). The values selected are open loop.
   9.5. PID LEARN. In setting up the PID constants the ACCG generates a sequence to get the first approximation of these constants. This has not proved to be particularly useful.
10. Table 1 lists the possible commands sent by the PC. The ACCG will respond using the same command as an echo to confirm the response.

Action	**Command Character**
ENQ	01_{hex}
SET PROCESS	02_{hex}
SET DEFAULT	03_{hex}
SET PARAMETER	04_{hex}
READ PARAMETER	05_{hex}
RECORD TARGET	06_{hex}
RESET MODULE	07_{hex}

Table 1- PC Commands

11. The ENQ command message carries no additional information. It purely triggers a response from the ACCG which includes the readings from the A/D converters and the status.

    11.1. Transmitted Message

11.1.1. Byte 0:- $3A_{hex}$
11.1.2. Byte 1:- 01_{hex}
11.1.3. Byte 2- 27:- 30_{hex}
11.1.4. Byte 28-31:- 16 bit checksum

11.2. Response Message (Standard Response)

11.2.1. Byte 0:- $3A_{hex}$
11.2.2. Byte 1:- 01_{hex}
11.2.3. Byte 2:- process number (see table 2)
11.2.4. Byte 3:- status flags
11.2.5. Byte 4-7:- 16 bit reading from Channel 0 (AC current input measurement)
11.2.6. Byte 8-11:- 16 bit reading from Channel 1
11.2.7. Byte 12-15:- 16 bit reading from Channel 2
11.2.8. Byte 16:- ACCG software version, most significant digit *e.g.* version **2**.1
11.2.9. Byte 17: ACCG software version, least significant digit *e.g.* version 2.**1**
11.2.10. Byte 16-27:- 30_{hex}
11.2.11. Byte 28-31:- 16 bit checksum

11.3. Possible Processes of the ACCG

11.3.1. The number will be an ASCII character

Process Number	**Description**
0	IDLE, output off
1	OPEN LOOP, output current at a fixed DA setting
2	PID, control loop- maintain output current
3	STEP, turn the output current high and low
5	PID LEARN, find the first iteration of the PID constants

Table 2: ACCG Processes

11.4. Status flags of the ACCG

11.4.1. The status flags are grouped together to create a 4 bit digit that is translated to an ASCII character.
11.4.2. Bit 0 (least significant bit):- Process Terminate. When the process is terminated and needs to be restarted this bit is set. *e.g.* If the AC current loop is opened once control has been started, the ACCG will not attempt to restart controlling the loop, but turn the output off and wait.

11.4.3. Bit 1:- AC current output unstable, do not use for calibration when set.
11.4.4. Bit 2:- Reserved
11.4.5. Bit 3:- Reserved

12. The SET PROCESS command allows a different process to be executed without changing the default process as stored in EEPROM.

12.1. Transmitted Message

12.1.1. Byte 0:- $3A_{hex}$
12.1.2. Byte 1:- 02_{hex}
12.1.3. Byte 2:- Process number as in table 2 (in ASCII)
12.1.4. Byte 3,4,5,6:- If byte 2 is OPEN LOOP, these 4 bytes represent the value 0-511 that the output current is set to. Otherwise the contents of all four bytes is $30_{hex.}$
If byte 2 is PID, then these 4 bytes represent the target value of the measured A/D reading on channel 0 (less than 4095).
12.1.5. Byte 7- 27:- 30_{hex}
12.1.6. Byte 28-31:- 16 bit checksum

12.2. Response Message (Standard Response)

12.2.1. Byte 0:- $3A_{hex}$
12.2.2. Byte 1:- 02_{hex}
12.2.3. Byte 2:- process number (see table 2)
12.2.4. Byte 3:- status flags (see table 3)
12.2.5. Byte 4-7:- 16 bit reading from Channel 0 (AC current input measurement)
12.2.6. Byte 8-11:- 16 bit reading from Channel 1
12.2.7. Byte 12-15:- 16 bit reading from Channel 2
12.2.8. Byte 16:- ACCG software version, most significant digit *e.g.* version **2**.1
12.2.9. Byte 17: ACCG software version, least significant digit *e.g.* version 2.**1**
12.2.10. Byte 16-27:- 30_{hex}
12.2.11. Byte 28-31:- 16 bit checksum

13. The SET DEFAULT command will force all the EEPROM parameters to their default values]

13.1. Transmitted Message

13.1.1. Byte 0:- $3A_{hex}$
13.1.2. Byte 1:- 03_{hex}
13.1.3. Byte 2- 27:- 30_{hex}
13.1.4. Byte 28-31:- 16 bit checksum

13.2. Response Message (Standard Response)

13.2.1. Byte 0:- $3A_{hex}$
13.2.2. Byte 1:- 03_{hex}
13.2.3. Byte 2:- process number (see table 2)

13.2.4. Byte 3:- status flags (see table 3)
13.2.5. Byte 4-7:- 16 bit reading from Channel 0 (AC current input measurement)
13.2.6. Byte 8-11:- 16 bit reading from Channel 1
13.2.7. Byte 12-15:- 16 bit reading from Channel 2
13.2.8. Byte 16:- ACCG software version, most significant digit *e.g.* version **2**.1
13.2.9. Byte 17: ACCG software version, least significant digit *e.g.* version 2.**1**
13.2.10. Byte 16-27:- 30_{hex}
13.2.11. Byte 28-31:- 16 bit checksum

14. The SET PARAMETER command allows the PC to set a particular address in the EEPROM to a particular value. Only one EEPROM address can be changed at a time. If a parameter exceeds 16 bits it must be programmed one address at a time.

14.1. Transmitted Message

14.1.1. Byte 0:- $3A_{hex}$
14.1.2. Byte 1:- 04_{hex}
14.1.3. Byte 2,3:- EEPROM address (see table x for details)
14.1.4. Byte 4-7:- EEPROM data (see table x for details)
14.1.5. Byte 8- 27:- 30_{hex}
14.1.6. Byte 28-31:- 16 bit checksum

14.2. Response Message (Standard Response)

14.2.1. Byte 0:- $3A_{hex}$
14.2.2. Byte 1:- 04_{hex}
14.2.3. Byte 2:- process number (see table 2)
14.2.4. Byte 3:- status flags (see table 3)
14.2.5. Byte 4-7:- 16 bit reading from Channel 0 (AC current input measurement)
14.2.6. Byte 8-11:- 16 bit reading from Channel 1
14.2.7. Byte 12-15:- 16 bit reading from Channel 2
14.2.8. Byte 16:- ACCG software version, most significant digit *e.g.* version **2**.1
14.2.9. Byte 17: ACCG software version, least significant digit *e.g.* version 2.**1**
14.2.10. Byte 16-27:- 30_{hex}
14.2.11. Byte 28-31:- 16 bit checksum

15. The READ PARAMETER command reads back the parameters in the EEPROM. The command specifies the start address. The ACCG responds with the contents of that address in the EEPROM and the 5 subsequent addresses. If the selected address is beyond the limit of valid data, information will be returned, although it is meaningless.

15.1. Transmitted Message

15.1.1. Byte 0:- $3A_{hex}$
15.1.2. Byte 1:- 05_{hex}

15.1.3. Bit 2,3:- EEPROM address (see table 3 for details)
15.1.4. Byte 4- 27:- 30_{hex}
15.1.5. Byte 28-31:- 16 bit checksum

15.2. Response Message

15.2.1. Byte 0:- $3A_{hex}$
15.2.2. Byte 1:- 05_{hex}
15.2.3. Byte 2,3: EEPROM start address
15.2.4. Byte 4-7:- EEPROM data
15.2.5. Byte 8-11: EEPROM data
15.2.6. Byte 12-15: EEPROM data
15.2.7. Byte 16-19: EEPROM data
15.2.8. Byte 20-23: EEPROM data
15.2.9. Byte 24-27: EEPROM data
15.2.10. Byte 28-31:- 16 bit checksum

16. The RECORD TARGET command forces the ACCG to take the current reading on channel 0 and save it in the EEPROM "TARGET" location for use in the control algorithm.

16.1. Transmitted Message

16.1.1. Byte 0:- $3A_{hex}$
16.1.2. Byte 1:- 06_{hex}
16.1.3. Byte 2- 27:- 30_{hex}
16.1.4. Byte 28-31:- 16 bit checksum

16.2. Response Message (Standard Response)

16.2.1. Byte 0:- $3A_{hex}$
16.2.2. Byte 1:- 06_{hex}
16.2.3. Byte 2:- process number (see table 2)
16.2.4. Byte 3:- status flags (see table 3)
16.2.5. Byte 4-7:- 16 bit reading from Channel 0 (AC current input measurement)
16.2.6. Byte 8-11:- 16 bit reading from Channel 1
16.2.7. Byte 12-15:- 16 bit reading from Channel 2
16.2.8. Byte 16:- ACCG software version, most significant digit *e.g.* version **2**.1
16.2.9. Byte 17: ACCG software version, least significant digit *e.g.* version 2.**1**
16.2.10. Byte 16-27:- 30_{hex}
16.2.11. Byte 28-31:- 16 bit checksum

17. The RESET MODULE command will force the ACCG to undergo a hardware reset (which will then reload all the constants from EEPROM) after sending the response to the message

17.1. Transmitted Message

17.1.1. Byte 0:- $3A_{hex}$
17.1.2. Byte 1:- 07_{hex}

17.1.3. Byte 2- 27:- 30_{hex}
17.1.4. Byte 28-31:- 16 bit checksum

17.2. Response Message (Standard Response)

17.2.1. Byte 0:- $3A_{hex}$
17.2.2. Byte 1:- 07_{hex}
17.2.3. Byte 2:- process number (see table 2)
17.2.4. Byte 3:- status flags (see table 3)
17.2.5. Byte 4-7:- 16 bit reading from Channel 0 (AC current input measurement)
17.2.6. Byte 8-11:- 16 bit reading from Channel 1
17.2.7. Byte 12-15:- 16 bit reading from Channel 2
17.2.8. Byte 16:- ACCG software version, most significant digit *e.g.* version **2**.1
17.2.9. Byte 17: ACCG software version, least significant digit *e.g.* version 2.**1**
17.2.10. Byte 16-27:- 30_{hex}
17.2.11. Byte 28-31:- 16 bit checksum

18. EEPROM parameters

EEPROM address	Parameter	Description	Default Value
00_{hex}	E_FREQ	AC current frequency (4194304*E_FREQ)/2 [22] For 50Hz E_FREQ=50 & for 400Hz E_FREQ=400	60 (60Hz)
01_{hex}	E_COARSE	Gain on ML2023. LSB=G0, D1=G1	3. G0=G1=1.
02_{hex}	E_AMPLITUDE	Default value of the variable resistor after reset 0-minimum, 511-maximum	511
03_{hex}	E_TARGET	The value to which the control algorithm will control. A 12 bit binary number	2048
04_{hex}	E_PROCESS	The process that the ACCG will execute after reset. Can be overwritten during operation without affecting EEPROM.	0=IDLE
05_{hex}	MARK_VALUE	The value of the variable resistor (an hence the output current) during first of the two states of the STEP function. 0-minimum, 511-maximum	511
06_{hex}	SPACE_VALUE	The value of the variable resistor (an hence the output current) during second of the two states of the STEP function. 0-minimum, 511-maximum	0
07_{hex}	STEP_REPEATS	Number of times the step function is repeated. 0=infinite, . 0-minimum, 255-maximum	50
08_{hex}	MARK_TIME	The time the MARK state is held. It is in multiples of 10msecs.	10 i.e. 100msec
09_{hex}	SPACE_TIME	The time the SPACE state is held. It	20

		is in multiples of 10msecs.	i.e. 200msec
$0A_{hex}$	KKP	Proportional Constant in PID loop	40_{hex}
$0B_{ex}$	KKD	Differential Constant in PID loop	00_{hex}
$0C_{hex}$	KKI	Integral Constant in PID loop	00_{hex}
$0D_{hex}$	PID_CALIBRATE_ STEP	The value of the variable resistor (an hence the output current) during the step generated to measure the PID constants in PID. 0-minimum, 511-maximum	255
$0E_{hex}$	OPERATIONAL_ MINIMUM	When the measured AC current drops below this value (4.x mA) the AC current loop is deemed to be open circuit	370_{hex}
$0F_{hex}$	Not used		1
10_{hex}	Not Used		1
11_{hex}	AD_SAMPLE_ DELAY	The time between A/D conversions while in the control loop. . It is in multiples of 10msecs.	20 i.e. 200msec
12_{hex}	E_DATE	Date of calibration. For storage purposes only	0
13_{hex}	E_MONTH	Month of calibration. For storage purposes only	0
14_{hex}	E_YEAR	Year of calibration. For storage purposes only	0
15_{hex}	RECOVER_DELAY	When an open circuit in the AC current loop is detected, the ACCG can decide that the process is over (RECOVER_DELAY=0) and wait for further commands from the host. Alternatively if this parameter is not zero the number represents the time (in multiples of 10mS) the process is inhibited before it restarts.	0

Table 3: ACCG Parameters

User Interface

The PC interface was developed using the Professional Edition of Visual Basic 4. This allowed the use of the serial communications features of the language.

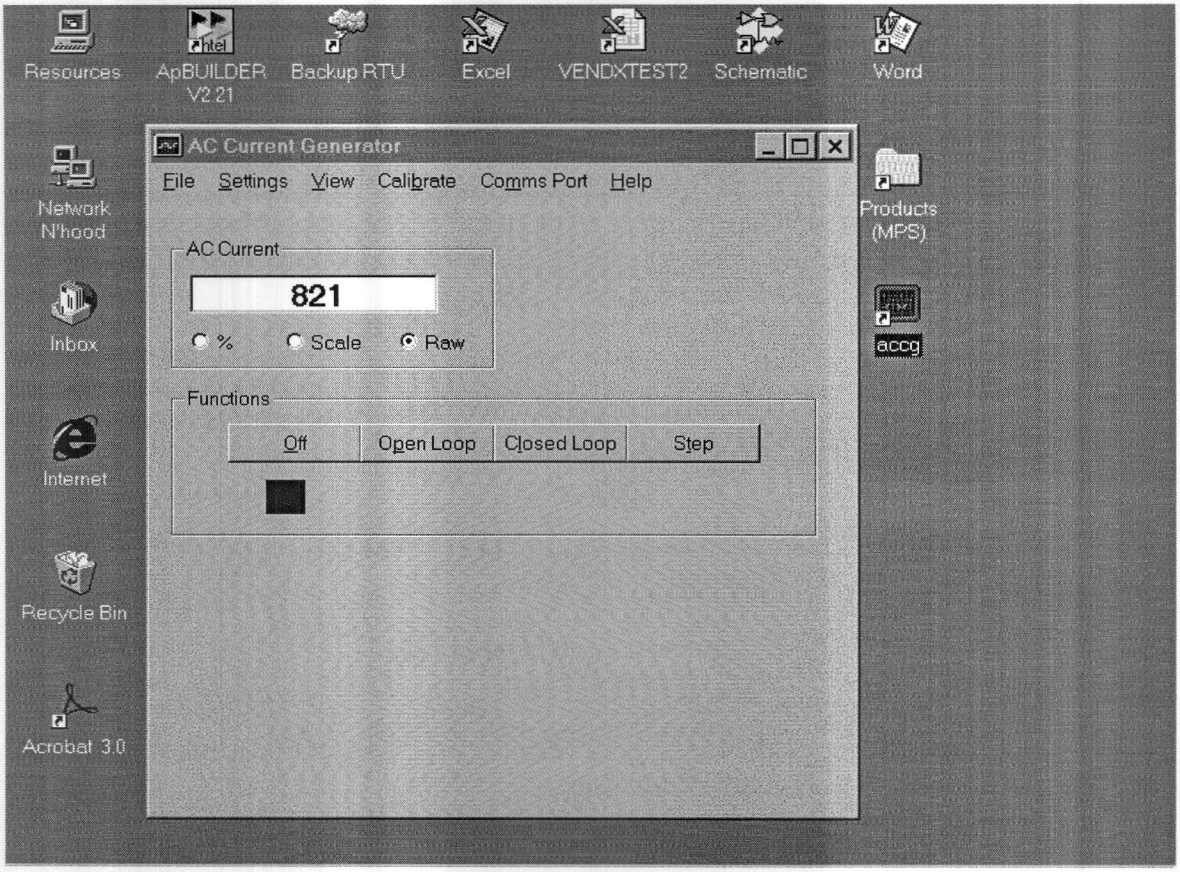
Perhaps the most difficult aspect of creating a program in Visual Basic is in fact documenting and describing it. I have provided all the files created in the hope that it will alleviate confusion.

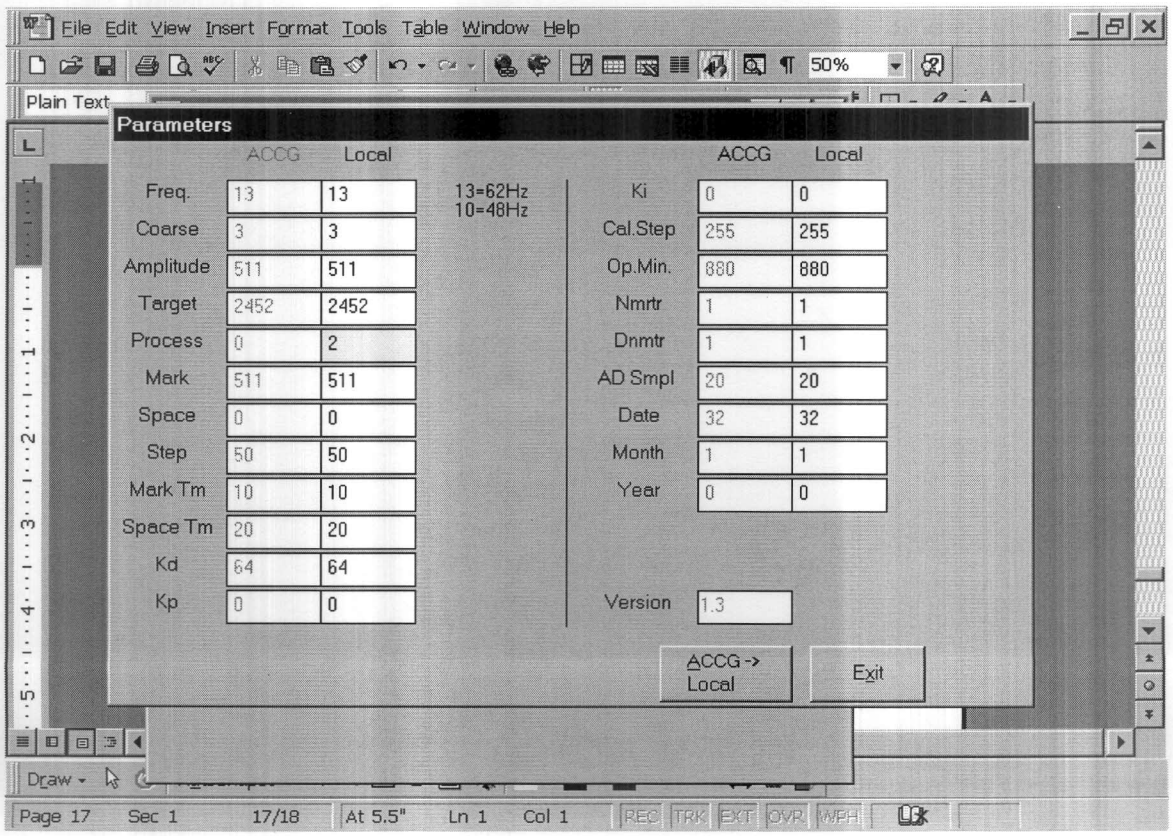
In terms of programming, an embedded programmer has to adjust to the concept of "event driven". In itself this is not so hard to understand or even adapt to since large portions of embedded design are event driven. However there are some cases where the event is not readily available and possible additional events added to generate a

"quasi event" in order to solve the problem. For instance, in this application the host generates a serial message and transmits it. If the ACCG is not connected no response is generated and hence no event. An additional conditional timer is then needed to check for this timeout condition.

I tried to generate the screen to resemble a piece of hardware I might have designed. There are pushbuttons and LEDs as well as an LCD display. However selecting one of the pushbuttons changes the mode and will present a number of contextual options. Real hardware does not have the luxury of "cloaking" and "unveiling" additional function buttons, so I have embraced the new approach to further my knowledge of the language.

A second screen allows the user to fetch all the parameters from the ACCG, then to modify them and save them back to the ACCG. It is also possible to calibrate the system for a given set of circumstances to allow a readout of AC current on the LCD display of the virtual instrument.


Resources
ApBUILDER V2.21
Backup RTU
Excel
VENDXTEST2
Schematic
Word
Network N'hood
Inbox
Internet
Recycle Bin
Acrobat 3.0
Products (MPS)
accg
AC Current Generator
File Settings View Calibrate Comms Port Help
AC Current
821
% Scale Raw
Functions
Off Open Loop Closed Loop Step


File Edit View Insert Format Tools Table Window Help
50%
Plain Text
Parameters
ACCG Local
Freq. 13 13
Coarse 3 3
Amplitude 511 511
Target 2452 2452
Process 0 2
Mark 511 511
Space 0 0
Step 50 50
Mark Tm 10 10
Space Tm 20 20
Kd 64 64
Kp 0 0
13=62Hz
10=48Hz
ACCG Local
Ki 0 0
Cal.Step 255 255
Op.Min. 880 880
Nmrtr 1 1
Dnmtr 1 1
AD Smpl 20 20
Date 32 32
Month 1 1
Year 0 0
Version 1.3
ACCG -> Local
Exit
Draw
Page 17 Sec 1 17/18 At 5.5" Ln 1 Col 1
REC TRK EXT OVR WPH

Bill of Materials

Semiconductors
U1: Atmel, AT89C55-12PC
U2: Dallas Semiconductor, DS1232L
U3: National Semiconductor, NM93C66N
U4: Micro Linear, ML2037CP
(Can be replaced by ML2037IS plus Aries 16-350000-10)
U5: Dallas Semiconductor, DS1267-010
U7: National Semiconductor, LM12CLK
U8: National Semiconductor, LM7805CT
U9: Linear Technology, LTC1294DCN
U10: Maxim, MX584KN
U11: Analog Devices, AD626AN
U12: Maxim, MAX232CPE
D1: Diode, 1N4148
CR1,2: Diode MR821
Q1: Transistor, BC546B

Resistors
R1,6: 10K ¼W 1%
R5: 29K4 ¼W 1%
R7: 200K ¼W 1%
R8,9: 12K ¼W 1%
R10,11,13,14: 249R ¼W 1%
R12,15: 0R
R17: 1M ¼W 5%
R18,19: 2K2 ¼W 5%
R20: 4K7 ¼W 5%

Capacitors
C1,2: 22pF, 10%, 50V, 0.1" spacing, ceramic
C3,4,7,13,16,21,22,25,27-30,37-40: 100nF, 10%, 50V, 0.1" spacing, ceramic
C5,24,32,33,35,36: 10uF, 10%, 35V, 0.2" spacing, tantalum
C6,34: 1uF, 10%, 35V, 0.2" spacing, tantalum
C11,12:1uF, 10%, 50V, 0.3" spacing, ceramic
C14,15,18,19,20,23: 330uF, 20%, 63V, 0.3" spacing, radial, electrolytic
C17: 1nF, 10%, 50V, 0.1" spacing, ceramic
C31: 22uF, 10%, 10V, 0.2" spacing, tantalum
C18,19: 3300uF, 20%, 25V, 0.3" spacing, radial, electrolytic

Sundry
P1: Weidmuller, LP 5.08/2/90, 159454 (or 159283)
P2,6: Weidmuller, LP 5.08/3/90, 159455 (or 159282)
P4: Weidmuller, SL 3.5/10/180, 161424
P4: Weidmuller, BL 3.5/10, 161020
P5: 9 way socket D-sub vertical
U7 Heatsink: Wakefield 641-K
(plus 6-32 screws, washers and nuts & thermal compound)
U8 Heatsink: Wakefield 273-AB
(plus 4-40 screw, washers and nut & thermal compound)
SW1: C&K EP11SD1CBE
X1: HC49, 11.0592 MHz

X2: HC49, 4.194304 MHz
J1: 3 way header, 0.1" spacing, plus shunt
J2: 2 way header, 0.1" spacing, plus shunt
PCB: Weidmuller Canada #601230
RS100 Housing: 414487 305mm
RS100 End Plate: 203323 (2 pieces)
RS100 Feet: 207455 (2 pieces)
RS100 Screw: 401942 (4 pieces)
Fan: (fix directly to heatsink, if producing high currents) TOYO USTF602012MW

Development Tools

CAD: Accel, V14
8051 Software: IAR, Embedded Workbench V2.01E
Emulator: Metalink, Icemaster-PE
Host Software Development: Microsoft, Visual Basic 4, Professional Edition

Conclusion

This project was never intended to become a commercial product. Some of the choices made were based on product and samples that we had in stock, as well as the development tools available to us. Except for the printed circuit board we did not want to invest more into the
project than time. In addition, since this was a "background" project the development was subjected to many stops and starts and took so long that one of the integrated circuits used made obsolete (the ML2023 DIP package is no longer available, but the SMD part combined with a transitionary board from Aries Inc will resolve the spares issue).

References

Software fault tolerance staves off the errors that besiege uP systems, Dick Jarret, Electronic Design, August 9, 1984
Designing Reliable Software for Automotive Applications, Barry Yarkoni and John Wharton, Intel Corp. AR-102
Fault-tolerant software in real-time single-chip microcontroller systems, N.Q. Burnham and C.F. Cowling, Electronic Components and Applications, Vol 6 No 1, Philips Corp.
Visual Basic, Programmer's Guide to Serial Communications, Richard Grier, Mabry Publishing
Serial port Complete, Jan Axelson, Lakeview Research
The Beginner's Guide to Visual Basic 4.0, Peter Wright, Wrox Press Ltd., 1995
Visual Basic 4 Secrets, Harold Davis, IDG Books 1996
Microsoft Visual Basic documentation (Programmer's Guide, Language Reference, and Professional Features)
Real Time Computer Control, An introduction, Stuart Bennet, Prentice Hall 1994